Introduction to Imitation Learning

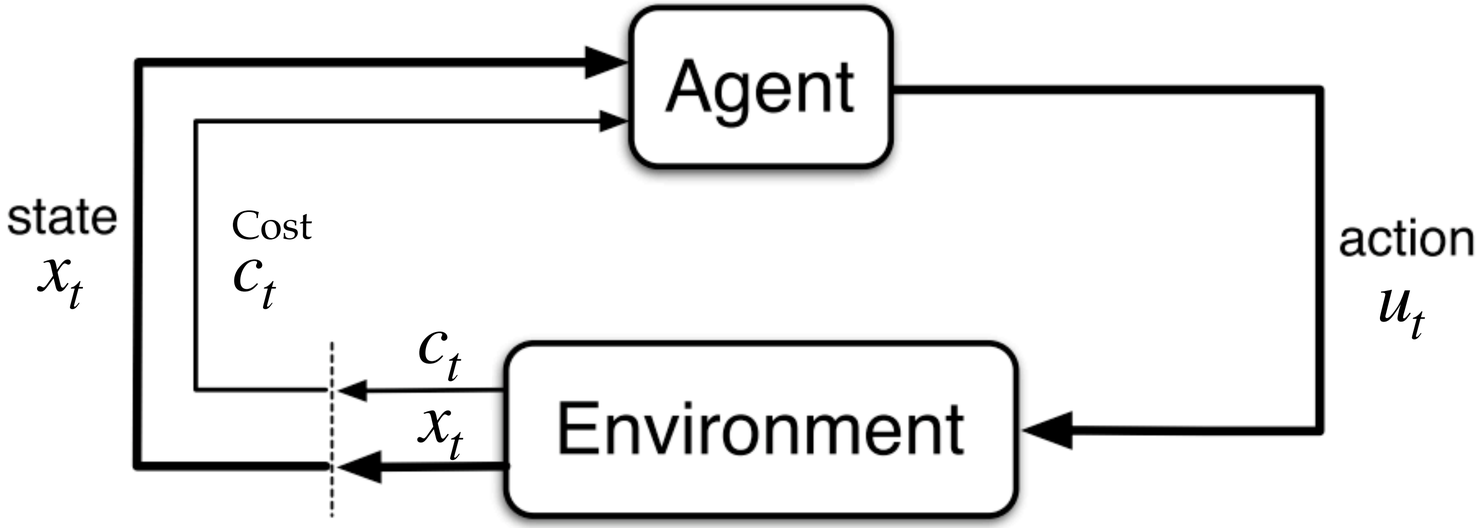
Matt Barnes

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

*Some content borrowed from Florian Shkurti, Yisong Yue and Hoang M. Le

Recap

- Markov Decision Processes are a very general class of models, which encompass planning and reinforcement learning.



Recap

"Markov" means that _____ captures all information about the history $x_1, x_2, \ldots, x_t$

The most recent state x_t

Recap

- The difference between planning and reinforcement learning is whether the _____ are known

transition model / dynamics / environment

Recap

- The three general methods for reinforcement learning are...

(1) Model-based

(2) Approximate dynamic programming

(3) Policy gradient

- (2) and (3) are both _____ methods

Model-free

Recap

The multi-armed bandit problem is reinforcement learning with __ state(s)

1

Recap

What is the fundamental trade-off in bandit (and reinforcement learning) problems?

Exploration vs. exploitation

Recap

Reward is equivalent to _____

Negative cost

Recap

The ε-greedy algorithm randomly explores with probability

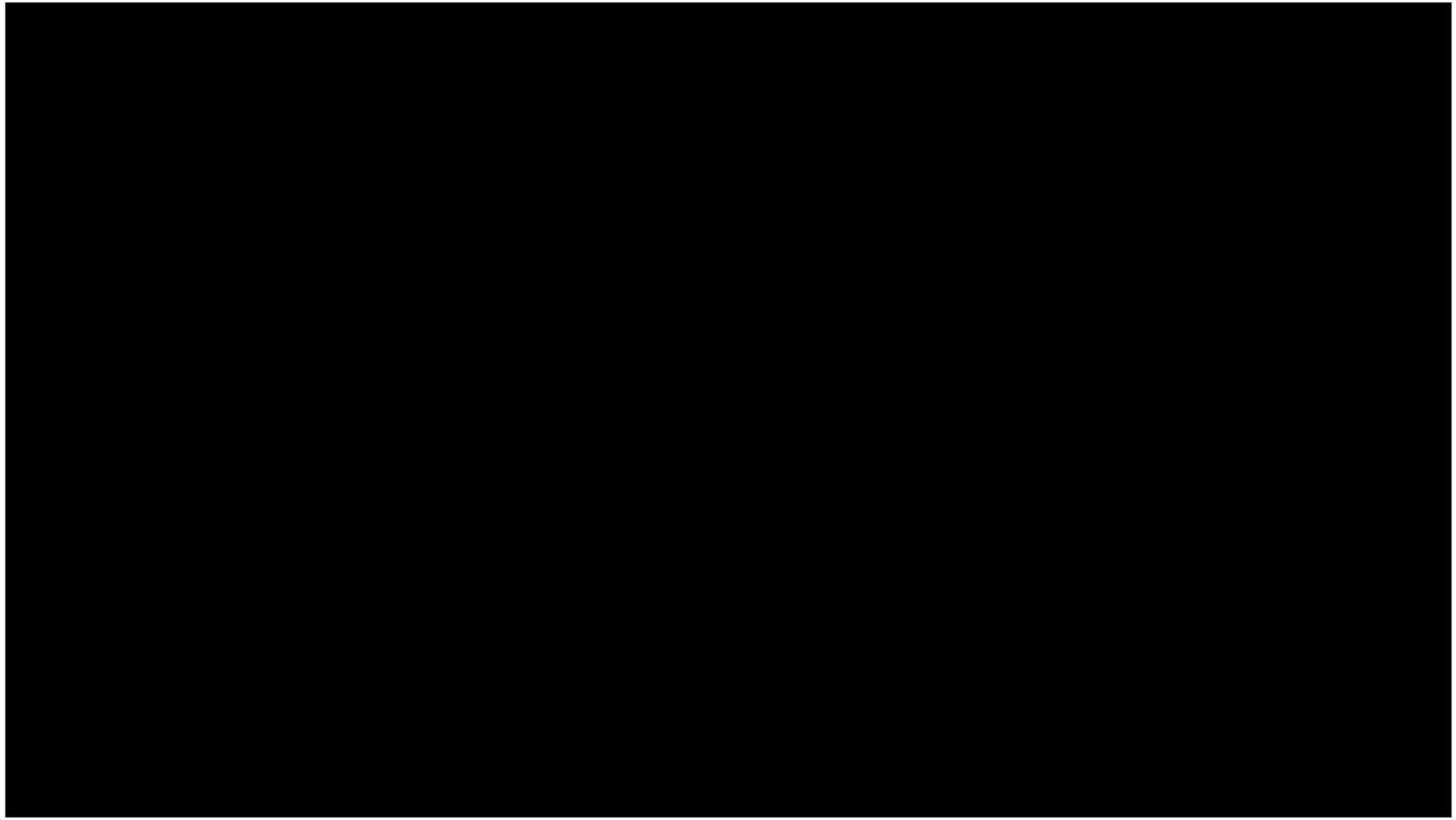
$\boldsymbol{\varepsilon}$

Recap

The UCB algorithm chooses actions according to the estimated reward, plus a bonus (i.e. confidence interval) which decreases with respect to _____

The number of times we try that action.

Today's lecture



Today's lecture

- Robots do not operate in a vacuum. They do not need to learn everything from scratch.
- Humans need to easily interact with robots and share our expertise with them.
- Robots need to learn from the behavior and experience of others, not just their own.

*Based off Florian Shkurti's lectures

Today's lecture

- Part 1: How can robots easily understand our objectives from demonstrations?

  Inverse reinforcement learning (IRL)

- Part 2: How can robots incorporate other's decision into their own?

  Behavior cloning (BC), interactive imitation learning (IL)

*Based off Florian Shkurti's lectures

Today's lecture

		Output		Inputs		
		Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstration
Part 1	Inverse RL	No	Yes	Yes	No	Yes
	Behavior Cloning	Yes (direct)	No	No	No	Yes
Part 2	Interactive IL	Yes (direct)	No	Yes	Yes	(Optional)
	GAIL	Yes (indirect)	No	Yes	No	Yes

*Partially based off slides from Yisong Yue and Hoang M. Le

Part 1: Inverse Reinforcement Learning (IRL)

- Setting: No reward function. For complex tasks, these can be hard to specify!

- Fortunately, we are given a set of demonstrations

$$D = \{\tau_1, \ldots, \tau_m\} = \{(x_0^i, u_0^i, x_1^i, u_1^i \ldots)\} \sim \rho_{\pi^*}$$

m trajectories

The state-action distribution of policy π^*

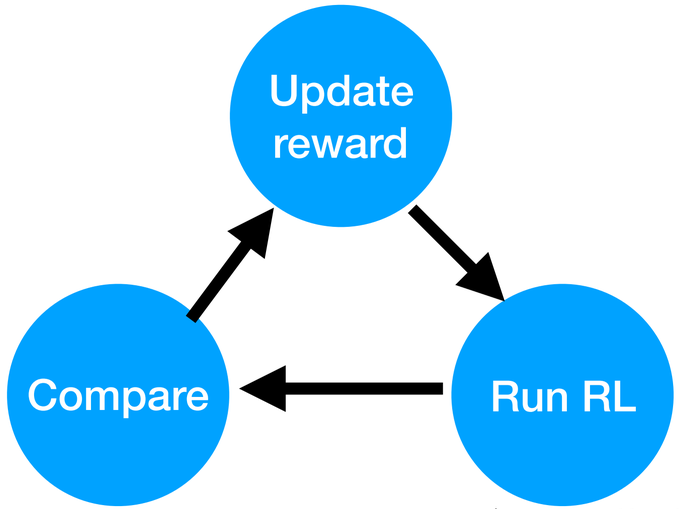
- **Goal**: Learn a reward function r^* such that

$$\pi^* = \textbf{argmax}_\pi \mathbb{E}_\pi \left[r^*(x, u) \right]$$

The high-level recipe

- Step 1: Learn a policy for the current reward function r
- Step 2: Update the reward function

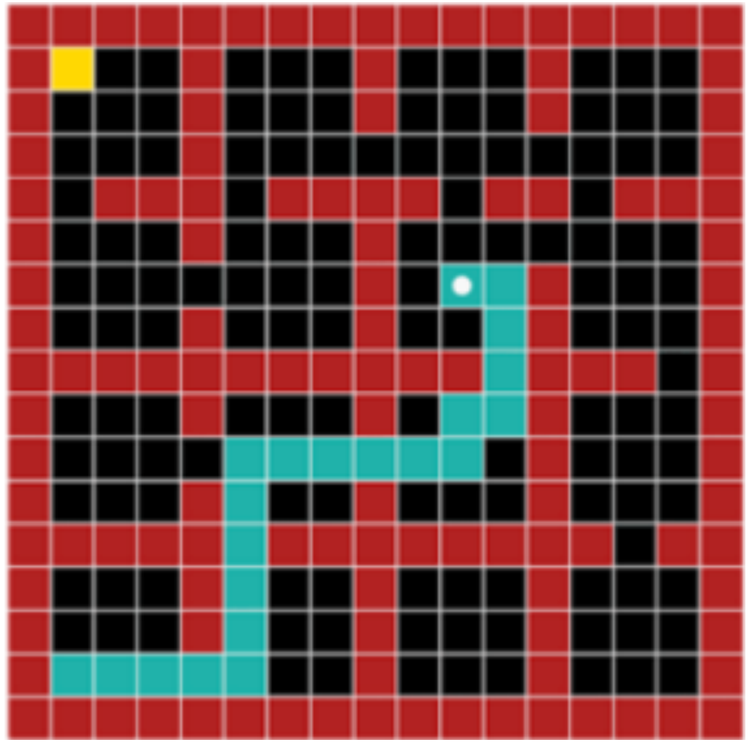
Repeat until policy trajectories are similar to demonstrations



*Partially based off slides from Yisong Yue and Hoang M. Le

Learning a reward function is an under defined problem

- Many reward functions correspond to the same policy



*Partially based off slides from Yisong Yue and Hoang M. Le

Learning a reward function is an under defined problem

- Let r^* be one solution, i.e. $\pi^* = \textbf{argmax}_\pi \mathbb{E}_\pi \left[r^*(x, u) \right]$
- Note that ar^* for any constant a is also a solution, since

$$\textbf{argmax}_\pi \mathbb{E}_\pi \left[ar^*(x, u) \right] = \textbf{argmax}_\pi \mathbb{E}_\pi \left[r^*(x, u) \right]$$

- In fact, $r^* = 0$ is always a solution, since every policy is optimal

Many different IRL approaches

- The reward function is linear [Abbeel & Ng 2004]
- Maximize the trajectory entropy, subject to a feature matching constraint [Ziebart et al., 2008]
- Maximum Margin Planning [Ratliff et al., 2006]

The linear reward function assumption reduces to feature expectation matching

- Assume $r(x) = \theta \cdot \phi(x)$ where $\phi(x)$ are the features of state x
- Then the value of a policy is linear in the expected features

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{H} \theta \cdot \phi(x_t)\right]$$

$$= \theta \cdot \mathbb{E}\left[\sum_{t=0}^{H} \phi(x_t)\right]$$

$$= \theta \cdot \mu(\pi)$$

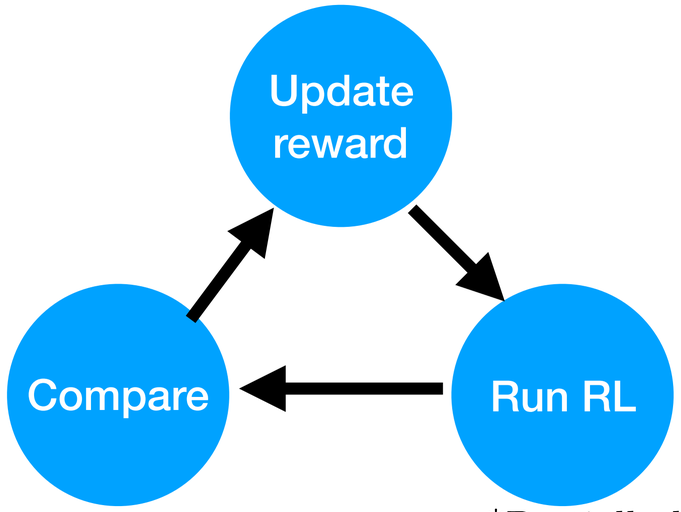
The expected features of the policy

New objective, matching the feature expectations

- Step 1: Learn a policy π for the current reward function and compute its feature expectation μ

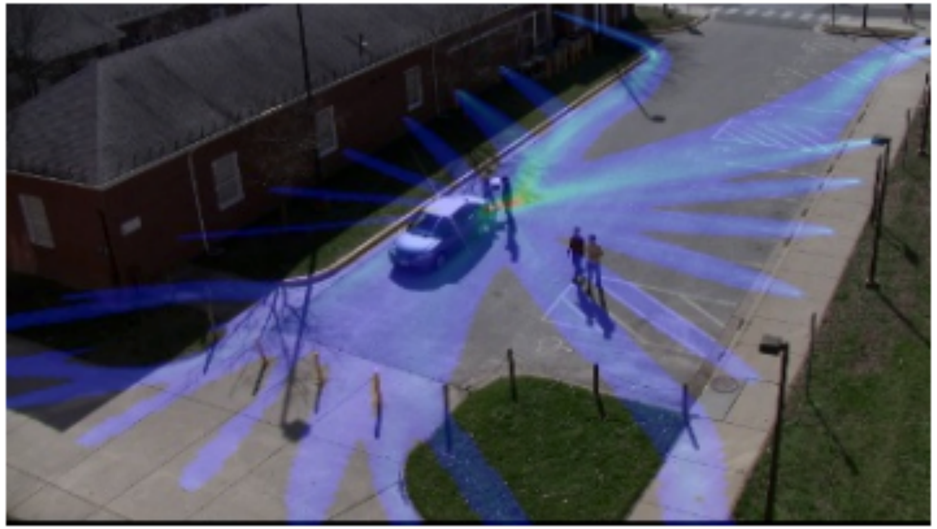
- Step 2: Update the reward function $\max_{||\theta|| \leq 1} \theta^T(\mu_\pi - \mu_{\pi^*})$

Repeat until feature expectations are close $||\mu_\pi - \mu_{\pi^*}||$



*Partially based off slides from Yisong Yue and Hoang M. Le

Using IRL to predict pedestrian intention



[Kitani et al., 2012]

Part 2: Directly learning a policy

		Output		Inputs		
		Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstration
Part 1	Inverse RL	No	Yes	Yes	No	Yes
	Behavior Cloning	Yes (direct)	No	No	No	Yes
Part 2	Interactive IL	Yes (direct)	No	Yes	Yes	(Optional)
	GAIL	Yes (indirect)	No	Yes	No	Yes

*Partially based off slides from Yisong Yue and Hoang M. Le

Behavior cloning

- Observe **pre-collected** expert demonstrations:

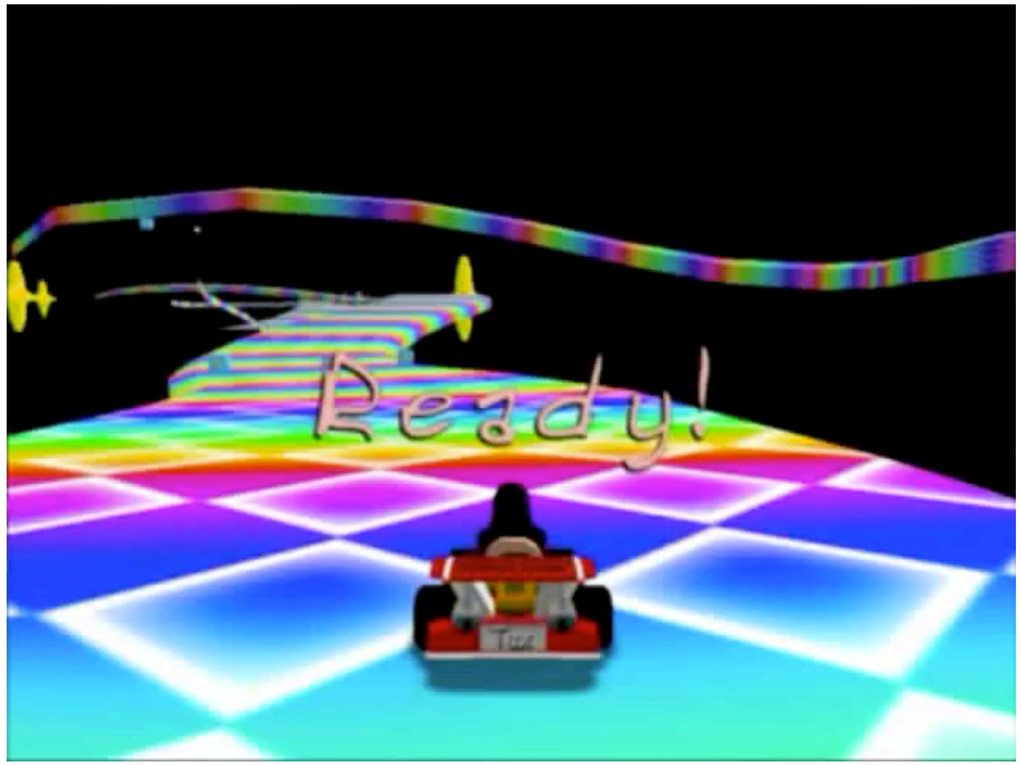
$$D = \{\tau_1, \ldots, \tau_m\} = \{(x_0^i, u_0^i, x_1^i, u_1^i \ldots)\} \sim \rho_{\pi^*}$$

The state-action distribution of policy π^*

- Learn a policy

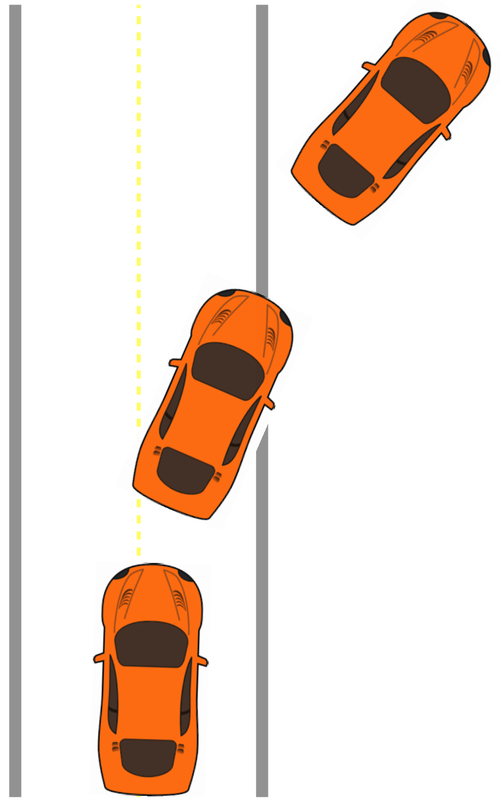
$$\hat{\pi} = \textbf{argmin}_\pi \sum_{i=0}^{n} \ell(u_i, \pi(x_i))$$

Some loss function



Behavior cloning suffers from compounding errors

- The good news: $\hat{\pi}$ will perform well on samples from ρ_{π^*}
- The really bad news: When we roll out policy $\hat{\pi}$ it will inevitably make some mistakes compared to π^*, and these errors could compound resulting in drastically different state action distributions ρ_{π^*} and $\rho_{\hat{\pi}}$



Behavior cloning suffers from compounding errors

- The good news: $\hat{\pi}$ will perform well on samples from ρ_{π^*}
- The really bad news: When we roll out policy $\hat{\pi}$ it will inevitably make some mistakes compared to π^*, and these errors could compound resulting in drastically different state action distributions ρ_{π^*} and $\rho_{\hat{\pi}}$

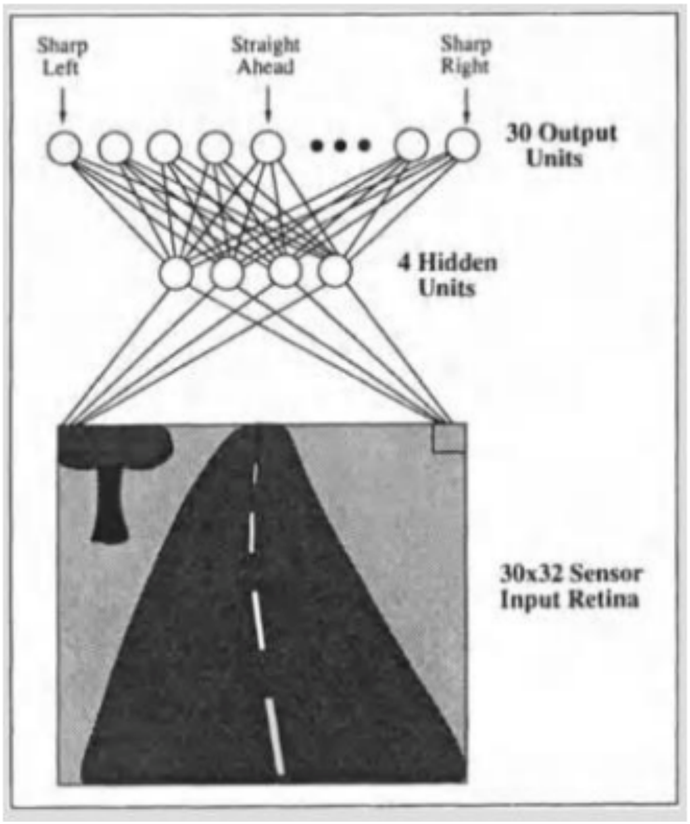
Behavior cloning suffers from compounding errors

Theorem (simplified) [Ross et al., 2011]. Let ε be the supervised learning error rate of $\hat{\pi}$. Then the cumulative reward of this policy is bounded by:

$$J(\hat{\pi}) \leq J(\pi^*) + T^2\epsilon$$

Errors compound with respect to the time horizon T

A history of covariate shift in imitation learning



30 x 32 pixels, 3 layer network, outputs steering command from approximately 5 minutes of training data per road type [Pomerleau 1992]

Navlab 1 (1986-1989) and Navlab 2 + ALVINN

A history of covariate shift in imitation learning

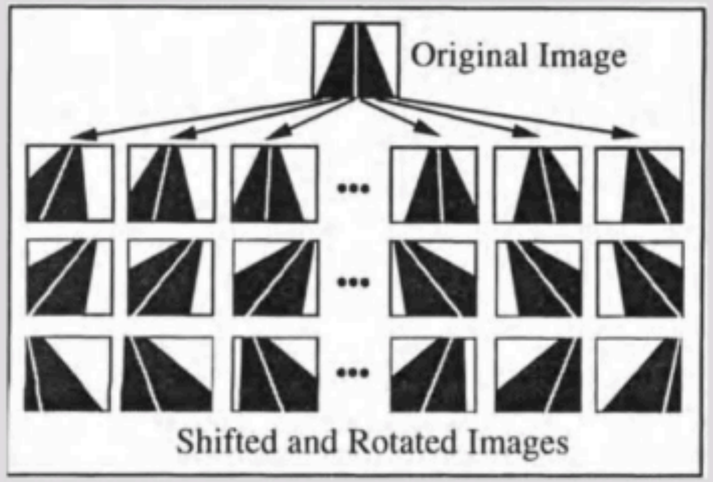
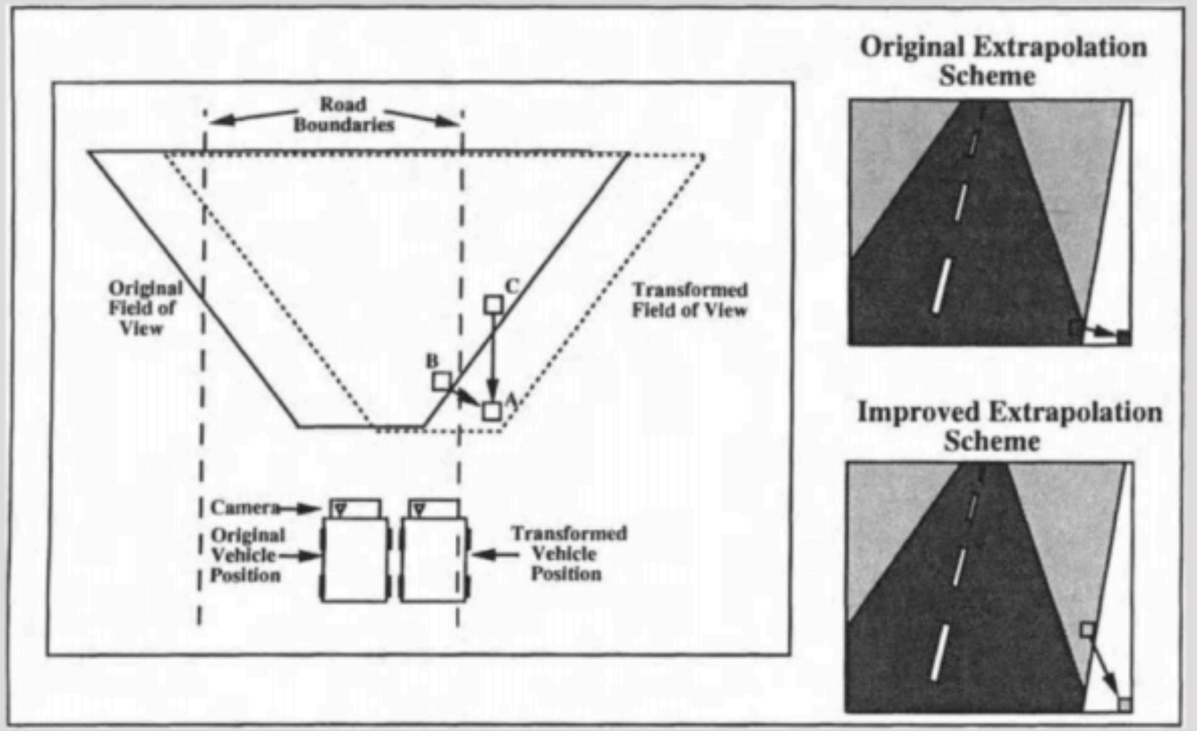


Figure 3.4: The single original video image is shifted and rotated to create multiple training exemplars in which the vehicle appears to be at different locations relative to the road.



[Pomerleau 1992]

Imitation Learning is not supervised learning

- Policy's actions affect future observations/data
- This is not the case in supervised learning

Imitation Learning

- Train/test data are not i.i.d.
- If expected hold-out error is ε, then expected test error after T decisions is up to

$$T^2\varepsilon$$

- Errors compound

Supervised Learning

- Train/test data are i.i.d.
- If expected hold-out error is ε, then expected test error after T decisions is order

$$T\varepsilon$$

- Errors are independent

*From Florian Shkurti

Interactive imitation learning

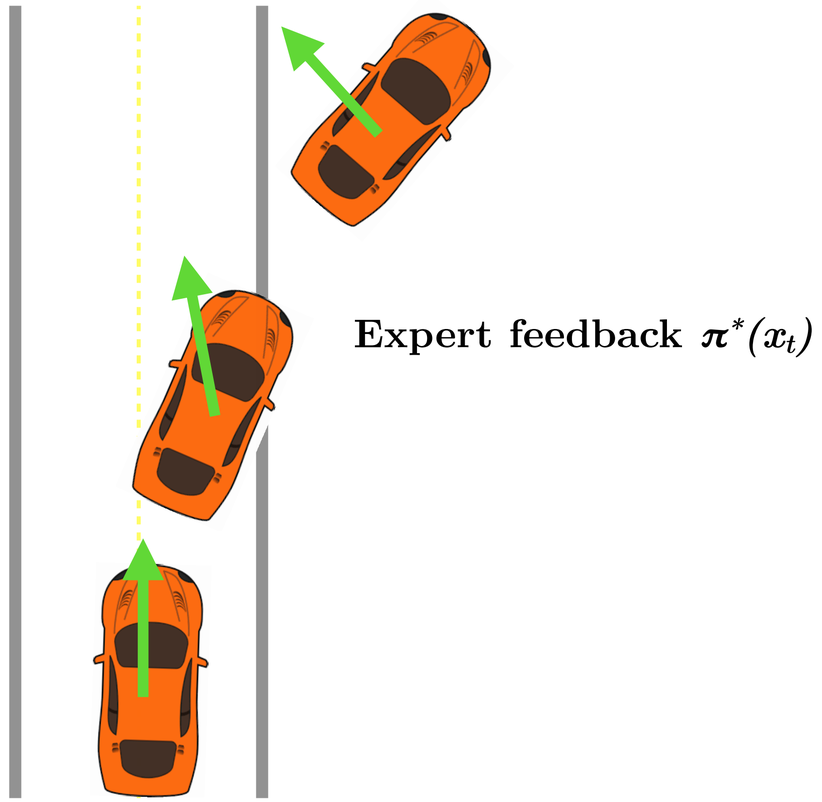
		Output		Inputs		
		Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstration
Part 1	Inverse RL	No	Yes	Yes	No	Yes
	Behavior Cloning	Yes (direct)	No	No	No	Yes
Part 2	**Interactive IL**	Yes (direct)	No	Yes	Yes	(Optional)
	GAIL	Yes (indirect)	No	Yes	No	Yes

*Partially based off slides from Yisong Yue and Hoang M. Le

Interactive feedback

- Roll-out any policy, and expert provides feedback for the current state

- In today's lecture, we'll consider the simple setting where this feedback takes the form of the expert's action $\boldsymbol{\pi^*(x_t)}$



The high-level recipe

- **Step 1**: Roll-out the current policy, collect expert feedback on the states it visits
- **Step 2**: Update the dataset and retrain the policy

Repeat

The Forward Training algorithm

Initialize T policies, $\pi_1, \ldots, \pi_T$

- Step 1: Roll-out policy $u_1 \sim \pi_1(s_1), u_2 \sim \pi_2(s_2), \ldots$

  and collect expert feedback $u_1^* = \pi^*(x_1), u_2^* = \pi^*(x_2), \ldots$

- Step 2: Update policies

Intuitively, each policy will have to learn to correct for the mistakes of earlier policies

The Forward Training algorithm

- Theorem (simplified) [Ross et al., 2011]. Let ε be the supervised learning error rate of $\hat{\pi}$. Then the cumulative reward of this policy is bounded by:

$$J(\hat{\pi}) \leq J(\pi^*) + uT\epsilon$$

A constant

Errors increase linearly with respect to the time horizon T, Same as supervised learning!

The downside: We have to learn T separate policies

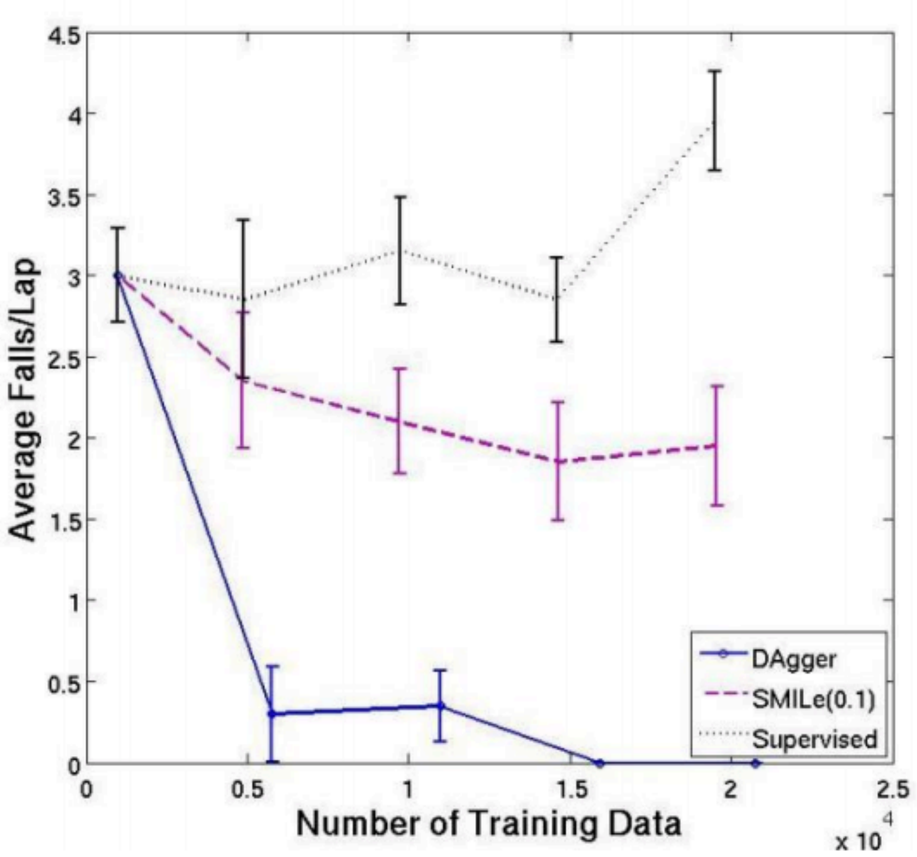
DAgger: Dataset Aggregation

- Learn only a single policy

Algorithm 1 DAgger

1: $D = \{(s, a)\}$ initial expert demonstrations
2: $\theta_1 \leftarrow$ train learner's policy parameters on D
3: **for** $i = 1...N$ **do**
4: Execute learner's policy π_{θ_i}, get visited states $S_{\theta_i} = \{s_0, ..., s_T\}$
5: Query the expert at those states to get actions $A = \{a_0, ..., a_T\}$
6: Aggregate dataset $D = D \cup \{(s, a) \mid s \in S_{\theta_i}, \ a \in A\}$ **Aggregate the data**
7: Train learner's policy $\pi_{\theta_{i+1}}$ on dataset D
8: Return one of the policies π_{θ_i} that performs best on validation set

DAgger: Dataset Aggregation



Ross et al, 2011

DAgger: Dataset Aggregation

Ross et al, 2011

GAIL: Generative Adversarial Imitation Learning

		Output		Inputs		
		Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstration
Part 1	Inverse RL	No	Yes	Yes	No	Yes
	Behavior Cloning	Yes (direct)	No	No	No	Yes
Part 2	Interactive IL	Yes (direct)	No	Yes	Yes	(Optional)
	GAIL	Yes (indirect)	No	Yes	No	Yes

GAIL: Generative Adversarial Imitation Learning

- Setting: Pre-collected expert demonstrations

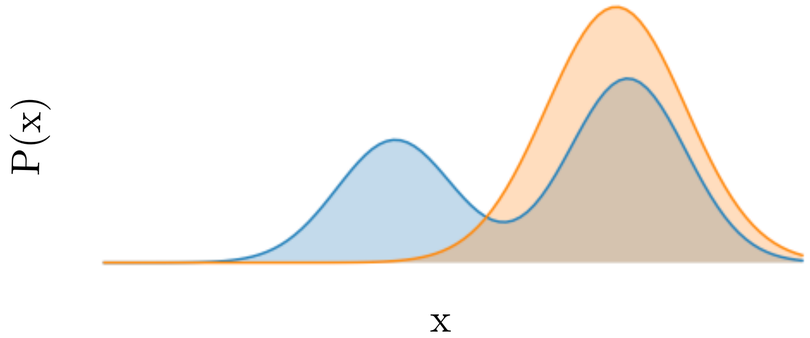
$$D = \{\tau_1, \ldots, \tau_m\} = \{(x_0^i, u_0^i, x_1^i, u_1^i \ldots)\} \sim \rho_{\pi^*}$$

- **Goal**: Minimize the *divergence* between ρ_{π^*} and $\rho_{\hat{\pi}}$

$$\hat{\pi} = \textbf{argmin}_{\pi} D(\rho_\pi || \rho_{\pi^*})$$

What's a divergence?

- Tells us how far apart two *distributions* are



- Given samples from ρ_{π^*} and $\rho_{\hat{\pi}}$, we can estimate the divergence (e.g. using [Nguyen et al., 2008])

GAIL: Generative Adversarial Imitation Learning

- The GAIL objective

$$\min_{\pi} \max_{f} \mathbb{E}_{x,u\sim\rho_{\pi^*}} \log(f(x,u)) + \mathbb{E}_{x,u\sim\rho_{\pi}} \log(1 - f(x,u))$$

RL problem

The "discriminator"

The divergence estimator

- **Step 1**: For the current π, maximize the discriminator to estimate the divergence

- **Step 2**: Update the policy using reinforcement learning.

Takeaways

- Expert demonstrations, and in particular expert feedback, can dramatically speed up policy training.
- Behavior Cloning suffers from compounding errors. This is not true for Forward Training or DAgger, which use interactive expert feedback.
- Inverse reinforcement learning allows us to learn an expert's reward function.